

This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

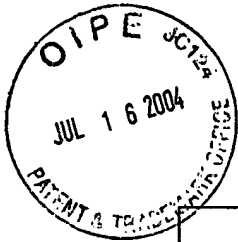
Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

**As rescanning documents *will not* correct images,
please do not report the images to the
Image Problem Mailbox.**



At 2154 IFW

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to: Commissioner for Patents, Mail Stop Appeal Brief - Patents, Box 1450, Alexandria, VA 22313-1450 on July 14, 2004.

Robert E. Malm

In re Application of:

RANDALL K. CUREY et al.

Serial Number: 09/821,537

Filing Date: 03/28/2001

For: PARTITIONED EXECUTIVE STRUCTURE
FOR REAL-TIME PROGRAMS

Group Art Unit: 2154

Examiner: FARHOOD MOSLEHI

Telephone: (703) 305-8646

SUBMISSION OF APPEAL BRIEF

Commissioner for Patents
Box 1450
Alexandria, VA 22313-1450

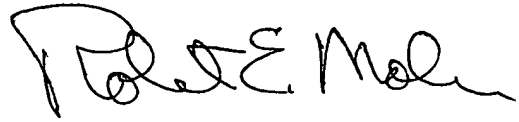
Dear Sir:

Transmitted herewith for filing is the appeal brief in triplicate in support of applicant's appeal to the Board of Patent Appeals and Interferences from the decision dated 03/26/04 of the examiner finally rejecting claims 1-49 of the application referenced above.

A check for the filing fee in the amount of \$330 is attached hereto.

The Commissioner is hereby authorized to charge any fees under 37 CFR 1.16 and 1.17 which may be required by this paper in excess of the above amount to Deposit Account No. 13-1239. (One additional copy of this Notice is enclosed herewith.)

Respectfully submitted,

A handwritten signature in black ink, appearing to read "R. E. Malm", with a stylized, cursive script.

Robert E. Malm
Reg. No. 34,662

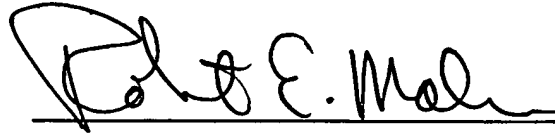
16624 Pequeno Place
Pacific Palisades, CA 90272
Tel. No: (310) 459-8728

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to: Commissioner for Patents, Mail Stop Appeal Brief - Patents, Box 1450, Alexandria, VA 22313-1450 on July 14, 2004.

Robert E. Malm



In re Application of:

RANDALL K. CUREY et al.

Serial Number: 09/821,537

Filing Date: 03/28/2001

For: PARTITIONED EXECUTIVE STRUCTURE
FOR REAL-TIME PROGRAMS

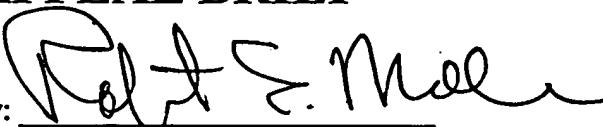
Group Art Unit: 2154

Examiner: FARHOOD MOSLEHI

Telephone: 703 305 8646

APPEAL BRIEF

Submitted by:



Robert E. Malm

Mailing Address:

16624 Pequeno Place
Pacific Palisades, CA 90272

Telephone:

(310) 459-8728

TABLE OF CONTENTS

TABLE OF AUTHORITIES	IV
INTRODUCTION	1
REAL PARTY IN INTEREST	9
RELATED APPEALS AND INTERFERENCES.....	9
STATUS OF CLAIMS	9
STATUS OF AMENDMENTS	9
SUMMARY OF INVENTION.....	10
ISSUES.....	18
GROUPING OF CLAIMS	18
ARGUMENT.....	20
I. WHETHER CLAIMS 1-2, 6-9, 21-22, 25-27, 31-34, AND 46-47 ARE UNPATENTABLE UNDER 35 U.S.C. § 102(B) AS HAVING BEEN ANTICIPATED BY MAITRA (U.S. 5,623,647).	20
CLAIMS 1, 25, AND 26.....	20
CLAIMS 2 AND 27.....	26
CLAIMS 6 AND 31.....	27
CLAIMS 7 AND 32.....	30
CLAIMS 8 AND 33.....	31
CLAIMS 9 AND 34.....	32
CLAIMS 21 AND 46.....	33
CLAIMS 22 AND 47.....	34
II. WHETHER CLAIMS 3-5, 18-20, 28-30, AND 43-45 ARE UNPATENTABLE UNDER 35 U.S.C. § 103(A) IN VIEW OF MAITRA (U.S. 5,623,647) AND JABLON ET AL. (U.S. 5,421,006).	36
CLAIMS 3 AND 28.....	36
CLAIMS 4 AND 29.....	38
CLAIMS 5 AND 30.....	40
CLAIMS 18 AND 43.....	41
CLAIMS 19 AND 44.....	43
CLAIMS 20 AND 45.....	44
III. WHETHER CLAIMS 10, 12, 35, AND 37 ARE UNPATENTABLE UNDER 35 U.S.C. § 103(A) IN VIEW OF MAITRA (U.S. 5,623,647) AND REZNAK (U.S. 6,223,201).	47
CLAIMS 10 AND 35.....	47
CLAIMS 12 AND 37.....	49
IV. WHETHER CLAIMS 11 AND 36 ARE UNPATENTABLE UNDER 35 U.S.C. § 103(A) IN VIEW OF MAITRA (U.S. 5,623,647), REZNAK (U.S. 6,223,201), AND DAVIDSON ET AL. (U.S. 6,292,934).	50
CLAIMS 11 AND 36.....	50
V. WHETHER CLAIMS 13, 14, 38, AND 39 ARE UNPATENTABLE UNDER 35 U.S.C. § 103(A) IN VIEW OF MAITRA (U.S. 5,623,647) AND YEN (U.S. 6,381,694).	51
CLAIMS 13 AND 38.....	51
CLAIMS 14 AND 39.....	52
VI. WHETHER CLAIMS 15-17 AND 40-42 ARE UNPATENTABLE UNDER 35 U.S.C. § 103(A) IN VIEW OF MAITRA (U.S. 5,623,647), YEN (U.S. 6,381,694), AND HARPER ET AL. (U.S. 6,629,266).	53
CLAIMS 15 AND 40.....	53

<i>CLAIMS 16 AND 41</i>	55
<i>CLAIMS 17 AND 42</i>	56
VII. WHETHER CLAIMS 23 AND 48 ARE UNPATENTABLE UNDER 35 U.S.C. § 103(A) IN VIEW OF MAITRA (U.S. 5,623,647) AND POTTER ET AL. (U.S. 5,014,327).....	57
<i>CLAIMS 23 AND 48</i>	57
VIII. WHETHER CLAIMS 24 AND 49 ARE UNPATENTABLE UNDER 35 U.S.C. § 103(A) IN VIEW OF MAITRA (U.S. 5,623,647) AND CIRCELLO ET AL. (U.S. 5,761,491).	59
<i>CLAIMS 24 AND 49</i>	59
CONCLUSIONS	61
APPENDIX.....	63

TABLE OF AUTHORITIES

CASES

Corning Glass Works v. Sumitomo Elec. U.S.A., Inc., 868 F.2d 1251, 1257, 9 USPQ2d 1962, 1966 (Fed. Cir. 1989)	22
In re Donaldson, 29 USPQ2d 1845 (Fed. Cir. 1994)	7
In re Stencel, 828 F.2d 751, 4 USPQ2d 1071 (Fed. Cir. 1987)	23
In re Vaeck, 947 F.2d 488, 20 USPQ2d 1438 (Fed. Cir. 1991)	36
Pac-Tec Inc. v. Amerace Corp., 903 F.2d 796, 801, 14 USPQ2d 1871, 1876 (Fed. Cir. 1990)	23
Pitney Bowes, Inc. v. Hewlett-Packagd Co., 182 F.3d 1298, 1305, 51 USPQ2d 1161, 1165 (Fed. Cir. 1999)	23
Richardson v. Suzuki Motor Co., 9 USPQ2d 1913, 1920 (Fed. Cir. 1989)	20
Verdegaal Bros. v. UnionOil Co. of California, 2 USPQ2d 1051, 1053 (Fed. Cir. 1987)	20

STATUTES

35 U.S.C. § 112, 6th paragraph	8
--------------------------------	---

OTHER AUTHORITIES

MPEP § 2111.01	7
MPEP § 2111.02	22, 23
MPEP § 2131	20
MPEP § 2142	36

INTRODUCTION

The invention that is the subject matter of this appeal is a method and apparatus for executing a plurality of software packages at a plurality of rates utilizing a common set of computational resources. The basis of the invention is to assign for execution each of the plurality of software packages to a particular sequence of time intervals, the sequence of time intervals assigned to one software package not overlapping the sequence of time intervals assigned to any other software package.

The invention can be analogized to a method for controlling access to a computer in a library. The librarian maintains a list of library patrons who wish to use the computer and simply assigns a different half-hour time increment each day to each patron on the list and this schedule continues from day to day. The librarian (L) applies for a patent for her method of controlling access to the computer. L's application contains a single claim:

A method for controlling access to a computer whereby each patron on a list is permitted to use the computer during each time interval of a sequence of time intervals assigned to the patron.

The examiner rejects L's claim on the basis of M's patent which discloses a method whereby the librarian assigns use of the computer in the order of registration of the patrons and also allocates a period of time that a patron may use the computer. When a patron's period of time expires, he is expected to relinquish the computer and the librarian notifies the next patron on the list that he may use the computer. If the patron completes his use of the computer before his period of time expires, he informs the librarian who notifies the next patron on the list that he may use the computer.

Are L's and M's methods the same? As a result of ignoring the words "assigned to the patron" in the claim, the examiner concluded that they are. Indeed, by ignoring the words "assigned to the patron" one could conclude that "each patron on a list is permitted to use the computer during

each time interval of a sequence of time intervals" simply by observing what has happened in the past.

The meaning of the claim is completely different when the words "assigned to the patron" are included. The act of "assigning" something for use occurs prior to use. Thus, in comparing the methods of L and M, one must look to the future and determine whether both methods involve assigning sequences of time intervals that extend from present time into the future.

L's method wherein a particular half-hour period (e.g. 0800-0830) occurring each day for weeks and months into the future are such a "sequence of time intervals". On the other hand, M's "just-in-time" assignment method assigns a time period for a patron to use the computer when the patron listed just before on the register finishes his use. M's method cannot assign a sequence of time intervals to a patron since M has no way of predicting when M's next turn to use the computer will occur. Thus, M does not disclose the L's invention as specified by the claim.

We return now to the subject matter of this appeal. This appeal has to do with a method and apparatus for executing a plurality of software packages at one or more rates utilizing a common set of computational resources. The basis of the invention is to generate non-overlapping sequences of time intervals and then to assign for execution each of the plurality of software packages to a particular sequence of time intervals.

Claim 1 of appellants' application claims a method for executing a plurality of software packages and reads as follows.

1. *A method [1] for repetitively executing a plurality of software packages at one or more rates, utilizing a common set of computational resources, the method comprising the steps:*

[2] generating a sequence of time intervals for each of the plurality of software packages, the time intervals belonging to one software package not overlapping the time intervals belonging to any other of the plurality of software packages;

[3] executing a plurality of software packages, each software package being executed during the time intervals of its sequence of time intervals.

The examiner cited Maitra in rejecting this claim and apparatus claims 25 and 26 which contain essentially the same limitations. Maitra discloses what one might call a "just-in-time" assignment method whereby a plurality of software packages are assigned numbers 1, 2, 3, . . . , N in the order of their execution by a microprocessor. Each software package is also assigned a "time quantum", the "assigned time quantum" being a length of time that the microprocessor may be engaged in executing the software package during a cycle of software-package executions. The "assigned time quantum" is not necessarily the execution time of the software package. The actual execution time can be either equal to or less than the "assigned time quantum".

The Maitra invention operates in the following way. The executing microprocessor calls up and executes successively software packages 1, 2, 3, . . . , N, 1, 2, 3, . . . N, . . . etc. A software package is executed to the point where the "assigned time quantum" is exhausted, at which point execution of the software package is suspended and execution of the next software package on the list begins. If the execution of the software package has blocked or finished before the "assigned time quantum" is exhausted, the microprocessor switches to the next software package on the list.

Since the time interval between the execution start times of successively-executed software packages is unknown, one cannot predict the time interval between the start times of successive executions of a particular software package. Maitra's just-in-time assignment process assigns a current time interval to a particular software package for execution by the microprocessor but is

incapable of assigning a sequence of time intervals extending from present time into the future to the particular software package.

Moreover, since the time interval between executions of a particular software package may vary from cycle to cycle, there is no way of associating a rate of execution with any of the software packages. Thus, Maitra does not disclose limitation [1] of claim 1, "a method for repetitively executing a plurality of software packages at one or more rates."

Claim 1 is a method "for repetitively executing a plurality of software packages at one or more rates." A particularly simple embodiment of this limitation is represented by the pattern ABCABCABCABC where A, B, and C denote three different software packages. The width of each letter denotes the time interval allotted for the execution of the particular software package identified by the letter. "Repetitively executing a plurality of software packages" is illustrated by the repeated appearance of ABC. The rate of execution of A is the reciprocal of the time interval between the A's in the ABCABCABC . . . pattern. Thus, the ABCABCABC . . . pattern illustrates "repetitively executing a plurality of software packages at one rate."

Is this a strained interpretation of "repetitively executing a plurality of software packages at one or more rates"? One would think not. Surely, the "rate" referred to in repetitively performing any action at a particular rate means the number of times the action is performed per unit time. And yet the examiner finds that Maitra's discussion of "changing the clock speed of the processor running two applications" is a disclosure of "repetitively executing a plurality of software packages at one or more rates." 03/26/04 Office Action, ¶ 62. This is like saying a race car repetitively circling a track at a rate of 20 track circles per hour is the same as changing the race car engine's rpm. Changing the engine's rpm affects the race-car circling rate but it is not the same thing.

It would appear that the examiner ignored "repetitively" in the claim language and then interpreted "executing a plurality of software packages at one or more rates" as meaning "executing a plurality of software packages WITH THE MICROPROCESSOR OPERATING at one or more rates." This is changing a claim limitation, not interpreting a claim limitation.

Since the time interval between the execution start times of successively-executed software packages is unknown in the Maitra invention, one cannot predict the time interval between the start times of successive executions of a particular software package. Maitra's just-in-time assignment process assigns a current time interval to a particular software package for execution by the microprocessor but is incapable of assigning a sequence of time intervals extending from present time into the future to the particular software package. Thus, Maitra does not disclose limitation [2] of claim 1:

[2] generating a sequence of time intervals for each of the plurality of software packages, the time intervals belonging to one software package not overlapping the time intervals belonging to any other of the plurality of software packages.

If Maitra does not disclose limitation [2], Maitra is unable to disclose limitation [3] which requires knowledge on the part of the microprocessor as to the sequence of time intervals assigned to each of the software packages to be executed:

[3] executing a plurality of software packages, each software package being executed during the time intervals of its sequence of time intervals.

Note that the execution of the software packages (limitation [3]) comes after the generating of sequences of time intervals and assignment of the sequences of time intervals to software packages (limitation [2]). A software package cannot be executed during the time intervals of its sequence

of time intervals unless the microprocessor knows which sequence of time intervals belongs to the software package.

This time ordering of steps implies that the sequences of time intervals, extending into the future, are identifiable and assignable BEFORE execution of any of the software packages.

In the Maitra invention, the assignment of a time interval for execution of a software package is accomplished on a just-in-time basis. When execution of one software package ends, the execution of the next software package on the list begins. There is no way of predicting when future executions will take place. And Maitra does not disclose anything pertaining to limitation [2], generating a sequence of time intervals for each software package and limitation [3], executing a software package during the time intervals of its sequence of time intervals.

Nonetheless, as a result of rather curious interpretations of limitations [2] and [3], the examiner concludes that Maitra discloses both limitations.

Insofar as limitation [2] is concerned, the examiner interprets the limitation as meaning several software packages can run simultaneously on a non-interfering basis, echoing Maitra's disclosure. 03/26/04 Office Action, ¶ 64.

With respect to limitation [3], the examiner ignores the presence of limitation [2] which provides a context for understanding limitation [3] and concludes that limitation [3], standing by itself, is disclosed by Maitra. 03/26/04 Office Action, ¶ 66. Although the schedule for executing software packages in the future is unknown in the Maitra invention, the sequence of time intervals that were used in executing a software package in the past is known with certainty. If one ignores the future (as one can do if one ignores limitation [2]), then Maitra seems to disclose limitation [3], "executing a plurality of software packages, each software package being executed during the time intervals of its sequence of time intervals." Certainly, "a plurality of software packages" were

executed by Maitra's microprocessor. And certainly, "each software package was executed during the time intervals of its sequence of time periods" if we interpret "its sequence of time periods" as being the just-in-time assignments of time period made by the microprocessor when execution of one software package ended and the execution of the next software package began. The problem with the examiner's analysis is that one cannot ignore the generating step of limitation [2] which results in "a sequence of time intervals for each of the plurality of software packages." This "sequence of time intervals for each of the plurality of software packages" provides the antecedent basis for "its sequence of time intervals" that appears in limitation [3].

The examiner's "interpretation" of the claim-1 limitations are typical of his "interpretation" of the limitations of the other 48 claims that he has rejected. Appellants believe that the claims are sufficiently unambiguous as to preclude interpretations such as the examiner's. However, recognizing that differences of opinion are always possible, appellants respectfully request the Board to resolve any questions concerning claim interpretation by consulting appellants' specification and drawings in accordance with *In re Donaldson*:

"During examination, the claims must be interpreted as broadly as their terms reasonably allow. . . . There is one exception and that is when an element is claimed using language falling under the scope of 35 U.S.C. 112, 6th paragraph (often broadly referred to as means or step plus function language). In that case the specification must be consulted to determine the structure, material, or acts corresponding to the function recited in the claim. *In re Donaldson*, 29 USPQ2d 1845 (Fed. Cir. 1994)(see MPEP § 2181 - § 2186)." MPEP § 2111.01.

The limitations of the method claims are all step-plus-function limitations (i.e. the steps do not specify acts sufficient to accomplish the specified functions) and the limitations of the apparatus claims are all means-plus-function limitations (i.e. the means do not specify structure sufficient to

Application Number: 09/821,537

P573C

Art Unit: 2154

accomplish the specified functions). The claims are clearly within the scope of 35 U.S.C. § 112, 6th paragraph.

REAL PARTY IN INTEREST

The real party in interest is: **NORTHROP GRUMMAN CORPORATION**
1840 Century Park East, 18th Floor
Los Angeles, CA 90067-2199

RELATED APPEALS AND INTERFERENCES

The application is a CIP of Application 09/572,298, filed 05/16/2000, which has been appealed from the examiner to the Board of Patent Appeals and Interferences.

STATUS OF CLAIMS

Claims 1-49 are pending in the application.

Claims 1-49 were rejected and all are subject to appeal.

STATUS OF AMENDMENTS

No amendments were requested subsequent to the examiner's final rejection of the claims.

SUMMARY OF INVENTION

1. A method for repetitively executing a plurality of software packages at one or more rates, utilizing a common set of computational resources, the method comprising the steps:

generating a sequence of time intervals for each of the plurality of software packages, the time intervals belonging to one software package not overlapping the time intervals belonging to any other of the plurality of software packages (Specification: p. 4, lines 5-11; p. 10, lines 18-22);

executing a plurality of software packages, each software package being executed during the time intervals of its sequence of time intervals (Specification: p. 4, lines 12-21).

2. The method of claim 1 wherein the plurality of software packages of the "executing" step includes only valid software packages (Specification: p. 11, lines 7-9), the method further comprising the step:

utilizing one or more tests to identify the software packages that are valid (Specification: p. 11, lines 9-10).

3. The method of claim 2 wherein one of the tests for validity is a one's complement checksum test of a software package's program memory (Specification: p. 11, lines 10-11).

4. The method of claim 2 wherein a software package is assigned its own dedicated memory region, one of the tests for validity being whether the address returned for the software package's initialization procedure lies within its dedicated memory region (Specification: p. 11, lines 11-12).

5. The method of claim 4 wherein one of the tests is whether the address is returned within a predetermined time (Specification: p. 11, lines 15-17).

6. The method of claim 2 wherein a software package is assigned its own dedicated memory region, the software package's dedicated memory region including a stack memory region and/or a heap memory region (Specification: p. 8, lines 5-7), one of the tests for validity being whether the stack memory range and/or the heap memory range assigned during the execution of the software package's initialization procedure and the various associated entry points lies within the software package's dedicated memory region (Specification: p. 11, lines 12-15).

7. The method of claim 6 wherein one of the tests is whether the stack memory range and/or the heap memory range and the various associated entry points are returned within a predetermined time (Specification: p. 11, lines 15-17).

8. The method of claim 1 wherein a software package is assigned its own dedicated memory region (Specification: p. 8, lines 5-7).

9. The method of claim 8 wherein the software package's dedicated memory region includes a stack memory region, a software package's stack residing in the software package's stack memory region (Specification: p. 8, lines 5-7).

10. The method of claim 1 wherein a software package includes background tasks as well as foreground tasks, the background tasks being performed after the foreground tasks have been completed (Specification: p. 12, lines 8-14).

11. The method of claim 10 wherein a background task is an infinite loop (Specification: p. 12, lines 10-12).

12. The method of claim 10 wherein the software package causes the power utilized in executing the software package to be minimized after completion of the background tasks (Specification: p. 12, lines 12-14).

13. The method of claim 1 wherein a failure in the execution of a software package causes information to be logged in a failure log (Specification: p. 12, lines 15-18).

14. The method of claim 13 wherein a failure in execution is linked to the software package that caused the failure (Specification: p. 12, lines 15-16).

15. The method of claim 13 wherein quality of performance in executing a software package is represented by one or more performance-quality parameters, values of the one or more performance-quality parameters being determined from the information logged in a failure log, the execution of a software package being subject to a plurality of execution options, an execution option being selected on the basis of one or more performance-quality parameter values (Specification: p. 12, lines 15-22).

16. The method of claim 15 wherein the plurality of execution options are user configurable (Specification: p. 12, lines 21-22).

17. The method of claim 15 wherein performance-quality parameters include the number of failures and/or the rate of failures for one or more classes of failures recorded in a software package's failure log (Specification: p. 12, lines 19-20).

18. The method of claim 1 wherein safety-critical software is placed in one or more separate partitions thereby isolating the safety-critical software from non-safety-critical software (Specification: p. 13, lines 1-2).

19. The method of claim 1 wherein each of the plurality of software packages is assigned its own memory block, a software package being enableable to read data only from zero or more memory blocks associated with other software packages, the zero or more memory blocks readable by a software package being either predetermined or determined during execution of the software packages in accordance with a set of one or more rules (Specification: p. 8, line 20 - p. 9, line 14).

20. The method of claim 1 wherein each of the plurality of software packages is assigned its own memory block, a software package being enableable to write data only to zero or more memory blocks associated with other software packages, the zero or more memory blocks writeable by a software package being either predetermined or determined during execution of the software packages in accordance with a set of one or more rules (Specification: p. 8, lines 20 - p. 9, line 6 and lines 15-18).

21. The method of claim 1 wherein an executive software package enforces the discipline that each software package executes only during the time intervals of its sequence of time intervals, the executive software package determining when the execution of a software package extends into a time interval belonging to the sequence of time intervals assigned to another software package and performs a remedial action (Specification: p. 10, line 18 - p. 11, line 6).

22. The method of claim 1 wherein the presence of those software packages that are present is detected (Specification: p. 11, lines 7-18).

23. The method of claim 1 wherein one or more of the plurality of software packages are independently compiled, linked, and loaded (Specification: p. 10, lines 7-8).

24. The method of claim 1 wherein a software package has its own stack, the software package's stack being selected prior to executing the software package (Specification: p. 11, lines 19-22).

25. Apparatus for practicing the method of claim 1 (Specification: p. 4, lines 5-21; p. 10, lines 18-22).

26. Apparatus for repetitively executing a plurality of software packages at a plurality of rates, the apparatus comprising:

a means for generating a sequence of time intervals for each of the plurality of software packages, the time intervals belonging to one software package not overlapping the time intervals belonging to any other of the plurality of software packages (Specification: p. 4, lines 5-11; p. 10, lines 18-22);

a means for executing a plurality of software packages, each software package being executed during the time intervals of its sequence of time intervals (Specification: p. 4, lines 12-21).

27. The apparatus of claim 26 wherein the plurality of software packages executed by the "executing" means includes only valid software packages (Specification: p. 11, lines 7-9), the apparatus further comprising:

a means for utilizing one or more tests to identify the software packages that are valid (Specification: p. 11, lines 9-10).

28. The apparatus of claim 27 wherein one of the tests for validity is a one's complement checksum test of a software package's program memory (Specification: p. 11, lines 10-11).

29. The apparatus of claim 27 wherein a software package is assigned its own dedicated memory region, one of the tests for validity being whether the address returned for the software package's initialization procedure lies within its dedicated memory region (Specification: p. 11, lines 11-12).

30. The apparatus of claim 29 wherein one of the tests is whether the address is returned within a predetermined time (Specification: p. 11, lines 15-17).

31. The apparatus of claim 27 wherein a software package is assigned its own dedicated memory region, the software package's dedicated memory region including a stack memory region and/or a heap memory region (Specification: p. 8, lines 5-7), one of the tests for validity being whether the stack memory range and/or the heap memory range assigned during the execution of the

software package's initialization procedure and the various associated entry points lies within the software package's dedicated memory region (Specification: p. 11, lines 12-15).

32. The apparatus of claim 31 wherein one of the tests is whether the stack memory range and/or the heap memory range and the various associated entry points are returned within a predetermined time (Specification: p. 11, lines 15-17).

33. The apparatus of claim 26 wherein a software package is assigned its own dedicated memory region (Specification: p. 8, lines 5-7).

34. The apparatus of claim 33 wherein the software package's dedicated memory region includes a stack memory region, a software package's stack residing in the software package's stack memory region (Specification: p. 8, lines 5-7).

35. The apparatus of claim 26 wherein a software package includes background tasks as well as foreground tasks, the background tasks being performed after the foreground tasks have been completed (Specification: p. 12, lines 8-14).

36. The apparatus of claim 35 wherein a background task is an infinite loop (Specification: p. 12, lines 10-12).

37. The apparatus of claim 35 wherein the software package causes the power utilized in executing the software package to be minimized after completion of the background tasks (Specification: p. 12, lines 12-14).

38. The apparatus of claim 26 wherein a failure in the execution of a software package causes information to be logged in a failure log (Specification: p. 12, lines 15-18).

39. The apparatus of claim 38 wherein a failure in execution is linked to the software package that caused the failure (Specification: p. 12, lines 15-16).

40. The apparatus of claim 38 wherein quality of performance in executing a software package is represented by one or more performance-quality parameters, values of the one or more performance-quality parameters being determined from the information logged in a failure log, the execution of a software package being subject to a plurality of execution options, an execution option being selected on the basis of one or more performance-quality parameter values (Specification: p. 12, lines 15-22).

41. The apparatus of claim 40 wherein the plurality of execution options are user configurable (Specification: p. 12, lines 21-22).

42. The apparatus of claim 40 wherein performance-quality parameters include the number of failures and/or the rate of failures for one or more classes of failures recorded in a software package's failure log (Specification: p. 12, lines 19-20).

43. The apparatus of claim 26 wherein safety-critical software is placed in one or more separate partitions thereby isolating the safety-critical software from non-safety-critical software (Specification: p. 13, lines 1-2).

44. The apparatus of claim 26 wherein each of the plurality of software packages is assigned its own memory block, a software package being enableable to read data only from zero or more memory blocks associated with other software packages, the zero or more memory blocks readable by a software package being either predetermined or determined during execution of the software packages in accordance with a set of one or more rules (Specification: p. 8, line 20 - p. 9, line 14).

45. The apparatus of claim 26 wherein each of the plurality of software packages is assigned its own memory block, a software package being enableable to write data only to zero or more memory blocks associated with other software packages, the zero or more memory blocks

writable by a software package being either predetermined or determined during execution of the software packages in accordance with a set of one or more rules (Specification: p. 8, line 20 - p. 9, line 6 and lines 15-18).

46. The apparatus of claim 26 wherein an executive software package enforces the discipline that each software package executes only during the time intervals of its sequence of time intervals, the executive software package determining when the execution of a software package extends into a time interval belonging to the sequence of time intervals assigned to another software package and performs a remedial action (Specification: p. 10, line 18 - p. 11, line 6).

47. The apparatus of claim 26 wherein the presence of those software packages that are present is detected (Specification: p. 11, lines 7-18).

48. The apparatus of claim 26 wherein one or more of the plurality of software packages are independently compiled, linked, and loaded (Specification: p. 10, lines 7-8).

49. The apparatus of claim 26 wherein a software package has its own stack, the software package's stack being selected prior to executing the software package (Specification: p. 11, lines 19-22).

ISSUES

- I. WHETHER CLAIMS 1-2, 6-9, 21-22, 25-27, 31-34, AND 46-47 ARE UNPATENTABLE UNDER 35 U.S.C. § 102(B) AS HAVING BEEN ANTICIPATED BY MAITRA (U.S. 5,623,647).
- II. WHETHER CLAIMS 3-5, 18-20, 28-30, AND 43-45 ARE UNPATENTABLE UNDER 35 U.S.C. § 103(A) IN VIEW OF MAITRA (U.S. 5,623,647) AND JABLON ET AL. (U.S. 5,421,006).
- III. WHETHER CLAIMS 10, 12, 35, AND 37 ARE UNPATENTABLE UNDER 35 U.S.C. § 103(A) IN VIEW OF MAITRA (U.S. 5,623,647) AND REZNAK (U.S. 6,223,201).
- IV. WHETHER CLAIMS 11 AND 36 ARE UNPATENTABLE UNDER 35 U.S.C. § 103(A) IN VIEW OF MAITRA (U.S. 5,623,647), REZNAK (U.S. 6,223,201) AND DAVIDSON ET AL. (U.S. 6,292,934).
- V. WHETHER CLAIMS 13-14 AND 38-39 ARE UNPATENTABLE UNDER 35 U.S.C. § 103(A) IN VIEW OF MAITRA (U.S. 5,623,647) AND YEN (U.S. 6,381,694).
- VI. WHETHER CLAIMS 15-17 AND 40-42 ARE UNPATENTABLE UNDER 35 U.S.C. § 103(A) IN VIEW OF MAITRA (U.S. 5,623,647), YEN (U.S. 6,381,694) AND HARPER ET AL. (U.S. 6,629,266).
- VII. WHETHER CLAIMS 23 AND 48 ARE UNPATENTABLE UNDER 35 U.S.C. § 103(A) IN VIEW OF MAITRA (U.S. 5,623,647) AND POTTER ET AL. (U.S. 5,014,327).
- VIII. WHETHER CLAIMS 24 AND 49 ARE UNPATENTABLE UNDER 35 U.S.C. § 103(A) IN VIEW OF MAITRA (U.S. 5,623,647) AND CIRCELLO ET AL. (U.S. 5,761,491).

GROUPING OF CLAIMS

Insofar as Issue I is concerned:

- Claims 1, 25, and 26 stand or fall together;
- Claims 2 and 27 stand or fall together;
- Claims 6 and 31 stand or fall together;
- Claims 7 and 32 stand or fall together;
- Claims 8 and 33 stand or fall together;

Claims 9 and 34 stand or fall together;
Claims 21 and 46 stand or fall together;
Claims 22 and 47 stand or fall together.

Insofar as Issue II is concerned:

Claims 3 and 28 stand or fall together;
Claims 4 and 29 stand or fall together;
Claims 5 and 30 stand or fall together;
Claims 18 and 43 stand or fall together;
Claims 19 and 44 stand or fall together;
Claims 20 and 45 stand or fall together.

Insofar as Issue III is concerned:

Claims 10 and 35 stand or fall together;
Claims 12 and 37 stand or fall together.

Insofar as Issue IV is concerned:

Claims 11 and 36 stand or fall together.

Insofar as Issue V is concerned:

Claims 13 and 38 stand or fall together;
Claims 14 and 39 stand or fall together.

Insofar as Issue VI is concerned:

Claims 15 and 40 stand or fall together;
Claims 16 and 41 stand or fall together;
Claims 17 and 42 stand or fall together.

Insofar as Issue VII is concerned:

Claims 23 and 48 stand or fall together.

Insofar as Issue VIII is concerned:

Claims 24 and 49 stand or fall together.

ARGUMENT

I. WHETHER CLAIMS 1-2, 6-9, 21-22, 25-27, 31-34, AND 46-47 ARE UNPATENTABLE UNDER 35 U.S.C. § 102(B) AS HAVING BEEN ANTICIPATED BY MAITRA (U.S. 5,623,647).

"A claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference." *Verdegaal Bros. v. UnionOil Co. of California*, 2 USPQ2d 1051, 1053 (Fed. Cir. 1987). (Cited in MPEP § 2131.)

"The identical invention must be shown in as complete detail as is contained in the . . . claim." *Richardson v. Suzuki Motor Co.*, 9 USPQ2d 1913, 1920 (Fed. Cir. 1989). (Cited in MPEP § 2131.)

CLAIMS 1, 25, AND 26

Claim 1 reads as follows:

1. *A method [1] for repetitively executing a plurality of software packages at one or more rates, utilizing a common set of computational resources, the method comprising the steps:*
 - [2] generating a sequence of time intervals for each of the plurality of software packages, the time intervals belonging to one software package not overlapping the time intervals belonging to any other of the plurality of software packages;*
 - [3] executing a plurality of software packages, each software package being executed during the time intervals of its sequence of time intervals.*

Apparatus claims 25 and 26 also contain limitations [1], [2], and [3].

The limitations shown in boldface are not disclosed by Maitra.

Limitation [1]

Maitra does not disclose *repetitively executing a plurality of software packages at one or more rates*. The concept of a software package "execution rate" is not to be found in Maitra. In fact, Maitra's disclosure indicates that there is no such thing as an "execution rate" for an application executed by the Maitra invention. Specifically, Maitra discloses how applications are executed in the following words:

"Typically the task scheduling unit 110 assigns each application a small unit of time, called a time quantum or time slice, when it is scheduled to run on the microprocessor 140. . . .

If the application has blocked or finished before the quantum has elapsed, the microprocessor switches to the next application." Maitra, col. 4, lines 32-44 (emphasis added).

To illustrate, consider three applications A, B, and C which are assigned by the task scheduling unit 110 to run according to a round-robin scheduling scheme. Maitra, col. 4, lines 32-42. Applications A, B, and C are assigned "time quanta" equivalent respectively to three letter time periods, five letter time periods, and two letter time periods as indicated in the following sequence.

. . . |AAABBBBBBCC|AAABBBBBBCC|AAABBBBBBCC|AAABBBBBBCC|. . .

The vertical separators denote the end of one round-robin cycle and the beginning of the next.

The portion of a "time quantum" utilized by an application depends on the task being performed by the application. For example, in a spread-sheet application, the execution of a "page compute" takes more time than the execution of "record data entries". Maitra, col. 7, lines 38-44.

As a result, the time interval between repeat executions of a given application is essentially indeterminate and an execution rate of the application cannot be specified.

To illustrate, let us assume that application B requires three letter time periods to finish during the first round-robin cycle, one letter time period to finish during the second round-robin cycle, five letter time periods to finish during the third round-robin cycle, and two letter time periods to finish during the fourth round-robin cycle. Maitra states that under such circumstances "[if] the application has blocked or finished before the quantum has elapsed, the microprocessor switches to the next application." Maitra, col. 4, lines 42-44. Thus, in this situation the above sequence becomes:

. . . |AAABBBCC|AAABCC|AAABBBBBBCC|AAABBBCC|. . .

Note how the cycle period continually changes. As a result, one cannot say that there is a particular execution rate for any of the applications as called for by limitation [1].

It is sometimes argued that a limitation appearing in the preamble of a claim such as limitation [1] is merely a statement of the intended use of the device. However, the Manual of Patent Examining Procedure emphasizes:

"The claim preamble must be read in the context of the entire claim. The determination of whether preamble recitations are structural limitations or mere statements of purpose or use 'can be resolved only on review of the entirety of the [record] to gain an understanding of what the inventors actually invented and intended to encompass by the claim.' *Corning Glass Works v. Sumitomo Elec. U.S.A., Inc.*, 868 F.2d 1251, 1257, 9 USPQ2d 1962, 1966 (Fed. Cir. 1989)." MPEP § 2111.02.

For example, the "for use" clause in "a bicycle for use in getting from one place to another" is merely a statement of use and does not imply any structural limitations on the bicycle. On the other hand, the "for use" clause in "a cell phone for use in the Verizon Wireless network" is not merely a statement of use but also implies certain structural limitations on the cell phone so that it has the capability of being used in the Verizon Wireless network.

In the case of applicants' *"a method [1] for repetitively executing a plurality of software packages at one or more rates"*, the "for use" clause, like the cell phone example, is not merely a statement of use but also implies limitations to the functions performed in the steps of the method.

In such cases, the Manual of Patent Examining Procedure states:

"Any terminology in the preamble that limits the structure of the claimed invention must be treated as a claim limitation. See, e.g., *Corning Glass Works v. Sumitomo Elec. U.S.A., Inc.*, 868 F.2d 1251, 1257, 9 USPQ2d 1962, 1966 (Fed. Cir. 1989); *Pac-Tec Inc. v. Amerace Corp.*, 903 F.2d 796, 801, 14 USPQ2d 1871, 1876 (Fed. Cir. 1990). See also *In re Stencel*, 828 F.2d 751, 4 USPQ2d 1071 (Fed. Cir. 1987)" MPEP § 2111.02.

One might ask whether applicants' *"a method [1] for repetitively executing a plurality of software packages at one or more rates"* asserts limitations that are not already fully and intrinsically set forth by the limitations in the body of the claim:

"If the body of a claim fully and intrinsically sets forth all of the limitations of the claimed invention, and the preamble merely states, for example, the purpose or intended use of the invention, rather than any distinct definition of any of the claimed invention's limitations, then the preamble is not considered a limitation and is of no significance to claim construction. *Pitney Bowes, Inc. v. Hewlett-Packard Co.*, 182 F.3d 1298, 1305, 51 USPQ2d 1161, 1165 (Fed. Cir. 1999)." MPEP § 2111.02.

Neither of the body elements of the claim at issue disclose either separately or in combination *"a method [1] for repetitively executing a plurality of software packages at one or more rates"*, and thus, these words represent an additional structural limitation which must be treated as a legitimate claim limitation and not merely "a statement of purpose or use."

The examiner argues that limitation [1] is disclosed by Maitra's disclosure of running different software packages with the microprocessor operating at different speeds. 03/26/04 Office Action, p. 12, ¶ 62. However, operating a microprocessor at different speeds is not the same as repetitively executing different software packages at the same or different rates. As we have pointed

out above, there is no rate of execution that can be established for any of Maitra's software packages, regardless of what speed at which the microprocessor is operated.

Limitation [2]

Maitra does not disclose *generating a sequence of non-overlapping time intervals for each of the plurality of software packages.*

Maitra discloses a system where "[a]t the end of the time slice, the process is suspended and the next scheduled process is readied for execution." Maitra, col. 4, lines 40-42. In the round-robin scheduling approach, a list of applications that need to be run is maintained and the task scheduling unit 110 runs each application in the order that it appears on the list. Maitra, col. 4, lines 37-40. Nothing is disclosed in Maitra concerning the "generating a sequence of non-overlapping time intervals for each of the plurality of software packages". If such a sequence had been generated, Maitra would necessarily base his selection of an application to be performed by first identifying which sequence of non-overlapping time intervals contained the upcoming time interval and then determining which application was associated with that sequence.

The examiner suggests that this limitation is disclosed by Maitra at col. 6, lines 20-34 which discloses a multitasking environment whereby several applications are run simultaneously. 03/26/04 Office Action, p. 12, ¶ 64. But there is nothing in this passage that suggests *generating a sequence of non-overlapping time intervals for each of the plurality of software packages.* This limitation requires that each of the plurality of software packages must be associated with an identifiable sequence of time intervals. A disclosure of a multitasking environment is not a disclosure of generating (i.e. bringing into existence) a plurality of sequences of non-overlapping time intervals

(one for each application to be executed) and associating each of the sequences with an application to be executed.

Limitation [3]

Maitra does not disclose *executing a plurality of software packages, each software package being executed during the time intervals of its sequence of time intervals.*

To disclose this limitation, Maitra's apparatus would have to (1) identify the beginning of the next time interval, (2) identify the sequence of time intervals of which the next time interval is a member, (3) identify the application associated with the identified sequence of time intervals, and (4) execute the identified application beginning with the start of the next time interval. Rather than performing this process, Maitra's task scheduling unit 110 simply waits for the presently-executing application to quit or finish processing and causes the next application on the list of applications to be executed. Maitra, col. 4, lines 37-44. Maitra's approach to selecting an application for execution has nothing to do with selecting an application based on its association with the sequence of time intervals of which the upcoming time interval is a member.

The examiner suggests that limitation [3] is disclosed by the passage in Maitra at col. 6, lines 39-43, which states that the microprocessor operates at full speed when a first software package is allowed to run and operates at half speed when a second software package is allowed to run. 03/26/04 Office Action, p. 12, ¶ 66. But this passage says nothing about identifying the application to be executed as the application associated with the sequence of time intervals of which the upcoming time interval is a member. And Maitra's apparatus does not generate the specified sequences of time intervals to provide a timing basis for switching from a presently-executing

application to the next. Instead, Maitra's apparatus simply begins execution of another application whenever the presently-executing application finishes.

Maitra does not disclose any of the boldface limitations of claim 1 and consequently did not anticipate applicants' claim-1 invention.

Claim 25, which depends from claim 1, was therefore also not anticipated by Maitra.

Claim 26 is a claim for apparatus that includes all of the limitations of method claim 1. Consequently, as in the case of appellants' claim-1 invention, appellants' claim-26 invention also was not anticipated by Maitra.

CLAIMS 2 AND 27

Claim 2 reads as follows:

2. *The method of claim 1 wherein [1] the plurality of software packages of the "executing" step includes only valid software packages, the method further comprising the step:*
- [2] utilizing one or more tests to identify the software packages that are valid.*

Apparatus claim 27 also contains limitations [1] and [2].

The limitations shown in boldface are not disclosed by Maitra.

Maitra does not require that *[1] the plurality of software packages of the "executing" step includes only valid software packages.*

Maitra does not disclose *[2] utilizing one or more tests to identify the software packages that are valid.*

Maitra does not disclose the limitations in boldface of claim 2, and consequently Maitra did not anticipate applicants' claim-2 and claim-27 inventions.

The examiner responded to the above statements by pointing out that "Maitra states that the clock scheduling unit determines the computing requirements of the application (e.g. col. 8, lines 60-64)." The examiner then concludes: "An invalid application would not be scheduled and no computing resources would be allocated." 03/26/04 Office Action, p. 13, ¶ 68.

The examiner's argument seems to be that determining the computing requirements of an application is the same as limitation [1]. But determining the computing requirements of an application is not the same as *the plurality of software packages of the 'executing' step includes only valid software packages*. Maitra never discusses whether the software packages to be executed are tested for validity.

With respect to limitation [2], the examiner makes the following statements: "A non-valid application would not have its configuration information being read into memory." (03/26/04 Office Action, p. 13, ¶ 70); "A non-valid application would not be have [sic] an associated configuration file." (03/26/04 Office Action, p. 13, ¶ 72); "running a benchmark evaluator [to determine computing requirements of an application] would be a test to determine an application validity." (03/26/04 Office Action, p. 13, ¶ 74). None of these conclusions by the examiner substitute for the lack of disclosure of limitation [2] by Maitra.

Maitra does not disclose the limitations in boldface of claim 2, and consequently Maitra did not anticipate applicants' claim-2 and claim-27 inventions.

CLAIMS 6 AND 31

Claim 6 reads as follows:

6. *The method of claim 2 wherein [1] a software package is assigned its own dedicated memory region, [2] the software package's dedicated memory region including a stack memory*

region and/or a heap memory region, [3] one of the tests for validity being whether the stack memory range and/or the heap memory range assigned during the execution of the software package's initialization procedure and the various associated entry points lies within the software package's dedicated memory region.

Apparatus claim 31 also contains limitations [1], [2], and [3].

Maitra does not disclose any of the limitations shown above in boldface.

The examiner cites Maitra, col. 7, lines 66-67, and col. 8, lines 1-20, (03/26/04 Office Action, p. 3, ¶ 9) as disclosing all of the limitations of claims 6 and 31, but it does not.

Maitra discloses a single memory for storing "a first plurality of processor executable instructions, a second plurality of processor executable instructions and a third plurality of processor executable instructions" (col. 8, lines 7-10) but says nothing about these three sets of executable instructions being stored in dedicated regions of memory.

Nor does Maitra say anything about "a software package's dedicated memory region including a stack memory region and/or a heap memory region."

Nor is the test of validity "whether the stack memory range and/or the heap memory range assigned during the execution of the software package's initialization procedure and the various associated entry points lies within the software package's dedicated memory region" disclosed in the Maitra passage.

The examiner's response to the above arguments were as follows.

The examiner states that "Maitra discusses memory 420 can store instructions or code that are part of application programs [*sic*]." The examiner then concludes: "Since these applications

programs can run at different clock cycles, hence different speeds, they would be stored in their own dedicated memory regions." 03/26/04 Office Action, p. 14, ¶¶ 76, 78.

The examiner's conclusion does not follow from the Maitra disclosure. The Maitra disclosure is not a disclosure of limitation [1], *a software package is assigned its own dedicated memory region*. Maitra never mentions dedicated memory assignments.

The examiner states that "Maitra shows that the application characterization unit 320 determines the computing requirements of an application." 03/26/04 Office Action, p. 14, ¶ 80. The examiner concludes that this is a disclosure of limitation [3], *one of the tests for validity being whether the stack memory range and/or the heap memory range assigned during the execution of the software package's initialization procedure and the various associated entry points lies within the software package's dedicated memory region*. 03/26/04 Office Action, p. 14, ¶ 79. It is not. There is simply no way that a person skilled in the art would consider "determining the computing requirements of an application" as being a disclosure of limitation [3].

The examiner disagrees with appellants' contention that limitation [2], *the software package's dedicated memory region including a stack memory region and/or a heap memory region*, is not disclosed by Maitra. 03/26/04 Office Action, p. 15, ¶ 83. The examiner argues that "Maitra discusses Application characterization unit 422 comprises the second plurality of processor executable instructions which is executed by 410 processor in the manner shown in FIG. 6. Clock programming unit 423 comprises the third plurality of processor executable instruction stored in memory which is executed by 410 processor." The examiner then concludes that "Each application has its own stack residing within its own application because applications are running at different speeds." 03/26/04 Office Action, p. 15, ¶ 84.

Here also, the examiner's conclusion does not follow from the quoted Maitra disclosure. And here also, no person skilled in the art would consider the quoted material as being a disclosure of limitation [2], *the software package's dedicated memory region including a stack memory region and/or a heap memory region.*

Maitra does not disclose the limitations in boldface and consequently Maitra did not anticipate applicants' claim-6 and claim-31 inventions.

CLAIMS 7 AND 32

Claim 7 reads as follows:

7. *The method of claim 6 wherein one of the tests is whether the stack memory range and/or the heap memory range and the various associated entry points are returned within a predetermined time.*

Apparatus claim 32 contains the same limitations as claim 7.

Maitra does not disclose the limitations of claims 7 and 32. The examiner cites Maitra, col. 3, lines 4-19, as disclosing the limitation of claims 7 and 32 (03/26/04 Office Action, p. 3, ¶ 15), but it does not.

The cited passage of Maitra discloses how the operating frequency of the microprocessor is adjusted to meet the computing requirements of the processes. Maitra says nothing about performing tests to determine the validity of the applications to be executed, and more specifically, says nothing about a test of validity being "whether the stack memory range and/or the heap memory range and the various associated entry points are returned within a predetermined time."

Maitra does not disclose the limitations of claims 7 and 32 and consequently Maitra did not anticipate applicants' claim-7 and claim-32 inventions.

The examiner disagrees with applicants' contention that Maitra does not disclose the limitations of claims 7 and 32. 03/26/04 Office Action, p. 15, ¶ 85. The examiner argues that "Maitra as part of the application characterization unit discusses the validity of the stack and or heap memory. The task switch detection unit 310 determines the applications scheduled by the operating system to be run by microprocessor during each time quantum by accessing information from task scheduling unit." 03/26/04 Office Action, p. 15, ¶ 86.

The examiner's quoted material from Maitra is not a disclosure of the claim-7 limitation, "wherein one of the tests [for validity] is whether the stack memory range and/or the heap memory range and the various associated entry points are returned within a predetermined time." There is nothing in Maitra concerning "stack memory range", "heap memory range", or "associated entry points being returned within a predetermined time".

Maitra does not disclose the limitations of claims 7 and 32 and consequently Maitra did not anticipate applicants' claim-7 and claim-32 inventions.

CLAIMS 8 AND 33

Claim 8 reads as follows:

8. *The method of claim 1 wherein a software package is assigned its own dedicated memory region.*

Apparatus claim 33 contains the same limitation as claim 8.

Maitra does not disclose the limitation of claims 8 and 33. The examiner cites Maitra, col. 7, line 66, through col. 8, line 20, as disclosing the limitation of claims 8 and 33 (03/26/04 Office Action, p. 3, ¶ 11), but it does not.

Maitra discloses a single memory for storing "a first plurality of processor executable instructions, a second plurality of processor executable instructions and a third plurality of processor executable instructions" (col. 8, lines 7-10) but says nothing about these three sets of executable instructions being stored in dedicated regions of memory.

Maitra does not disclose the limitation of claims 8 and 33 and consequently Maitra did not anticipate applicants' claim-8 and claim-33 inventions.

The examiner did not respond to the above arguments in his 03/26/04 Office Action.

CLAIMS 9 AND 34

Claim 9 reads as follows:

9. *The method of claim 8 wherein the software package's dedicated memory region includes a stack memory region, a software package's stack residing in the software package's stack memory region.*

Apparatus claim 34 contains the same limitation as claim 9.

Maitra does not disclose the limitation of claims 9 and 34. The examiner cites Maitra, col. 7, line 66, through col. 8, line 20, as disclosing the limitation of claims 9 and 34 (03/26/04 Office Action, p. 3, ¶ 13), but it does not.

Maitra says nothing about the software package's dedicated memory region including a stack memory region nor does it say anything about a software package's stack residing in the software package's stack memory region.

Maitra does not disclose the limitations of claims 9 and 34 and consequently Maitra did not anticipate applicants' claim-9 and claim-34 inventions.

The examiner did not respond to the above arguments in his 03/26/04 Office Action.

CLAIMS 21 AND 46

Claim 21 reads as follows:

21. *The method of claim 1 wherein an executive software package enforces the discipline that each software package executes only during the time intervals of its sequence of time intervals, the executive software package determining when the execution of a software package extends into a time interval belonging to the sequence of time intervals assigned to another software package and performs a remedial action.*

Apparatus claim 46 contains the same limitations as claim 21.

Maitra does not disclose the limitations of claims 21 and 46. The examiner cites Maitra, col. 4, lines 22-35, as disclosing the limitation of claims 21 and 46 (03/26/04 Office Action, p. 4, ¶ 17), but it does not.

The cited passage of Maitra discloses how the "[t]ask scheduling unit 110 is coupled to microprocessor 140 and schedules the applications which are run by the microprocessor 140." As Maitra points out, the execution of an application is suspended at the end of its assigned "time slice" (col. 4, lines 40-42). The Maitra invention does not allow the execution of an application to extend

into another application's assigned "time slice". This is not a disclosure of the limitations of claims 21 and 46 which envision the possibility of the execution of a software package extending into another software package's assigned time interval and requiring a remedial action.

Maitra does not disclose the limitations of claims 21 and 46 and consequently Maitra did not anticipate applicants' claim-21 and claim-46 inventions.

The examiner's response to the above arguments was to state that "Maitra discusses priorities that can be assigned internally and externally to time slices, **once a priority changes one application can be executed in another applications time slice.** e.g. col. 4, lines 40-55." 03/26/04 Office Action, p. 16, ¶ 88. (In the interests of accuracy, the boldface portion of the examiner's response does not appear in the Maitra passage cited by the examiner.)

The subject matter of claims 21 and 46 has nothing to do with assigning priorities as discussed in the Maitra passage cited by the examiner.

Maitra does not disclose the limitations of claims 21 and 46 and consequently, Maitra did not anticipate applicants' claim-21 and claim-46 inventions.

CLAIMS 22 AND 47

Claim 22 reads as follows:

22. *The method of claim 1 wherein the presence of those software packages that are present is detected.*

Apparatus claim 47 contains the same limitation as claim 22.

Maitra does not disclose the limitations of claims 22 and 47. The examiner cites Maitra, col. 4, lines 22-35, as disclosing the limitation of claims 22 and 47 (03/26/04 Office Action, p. 4, ¶ 19), but it does not.

The cited passage of Maitra discloses that the "[t]ask scheduling unit 110 is coupled to microprocessor 140 and schedules the applications which are run by the microprocessor 140, the order in which the applications are run, and the amount of CPU time each application receives." Maitra discloses that scheduling unit 110 schedules THE APPLICATIONS WHICH ARE RUN BY THE MICROPROCESSOR 140. This passage says nothing about DETECTING THE PRESENCE OF APPLICATIONS TO BE RUN.

Maitre discloses that "the task scheduler 110 maintains a list of applications to be run" (col. 4, lines 37-38) but maintaining a list of applications to be run is not the same as DETECTING THE PRESENCE OF APPLICATIONS TO BE RUN.

Maitra does not disclose the limitation of claims 22 and 47 and consequently Maitra did not anticipate applicants' claim-22 and claim-47 inventions.

The examiner's response to the above arguments was "[k]eeping a list of applications shows whether an application is present or not." 03/26/04 Office Action, p. 16, ¶ 90. This is like arguing that the telephone directory shows whether a person is still a subscriber or not. It does not. If one wishes to determine whether a person is still a subscriber, one must call the number in the directory and make this determination from the answer one receives. The Maitra disclosure would suggest trusting the directory— NOT making the call and detecting the presence of the particular subscriber listed in the telephone directory.

**II. WHETHER CLAIMS 3-5, 18-20, 28-30, AND 43-45 ARE
UNPATENTABLE UNDER 35 U.S.C. § 103(A) IN VIEW OF MAITRA (U.S.
5,623,647) AND JABLON ET AL. (U.S. 5,421,006).**

Three basic criteria must be satisfied to establish *prima facie* obviousness:

"The legal concept of *prima facie* obviousness is a procedural tool of examination which applies broadly to all arts. . . . The examiner bears the initial burden of factually supporting any *prima facie* conclusion of obviousness. If the examiner does not produce a *prima facie* case, the applicant is under no obligation to submit evidence of nonobviousness. . . . To establish a *prima facie* case of obviousness, three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. The teaching or suggestion to make the claimed combination and the reasonable expectation of success must both be found in the prior art, and not based on applicant's disclosure. *In re Vaeck*, 947 F.2d 488, 20 USPQ2d 1438 (Fed. Cir. 1991)." MPEP § 2142.

CLAIMS 3 AND 28

Claim 3 reads as follows:

3. *The method of claim 2 wherein one of the tests for validity is a one's complement checksum test of a software package's program memory.*

Apparatus claim 28 contains the same limitation as claim 3.

Neither of the cited references discloses the limitation of claim 3. The passage in Jablon et al. cited by the examiner (col. 5, lines 4-18) (03/26/04 Office Action, p. 5, ¶ 23) describes the use of checksums to validate "extension programs" to the BIOS program which resides in the read-only memory of a PC. Jablon et al. says nothing about applying checksums in general to a software package's program memory. Nor does Jablon et al. say anything about the use of one's complement checksums in testing the validity of a software package.

Even if Jablon et al. did disclose applicants' limitation, there is no motivation to be found in either Maitra or Jablon et al. for a person skilled in the art including such a limitation in Maitra's invention. Certainly Maitra expresses no concern as to the need for testing the validity of the applications that his invention is intended to execute. And Jablon et al. actually teaches away from the use of checksums by stating that (col. 5, lines 15-18) the "storage protection method [of U.S. Pat. No. 5,022,077] shares the architectural weakness of most software-controlled protection schemes on the PC."

Thus, Maitra and Jablon et al. in combination do not meet two of the basic criteria for establishing *prima facie* obviousness of claims 3 and 28. Neither of the references discloses applicants' one's complement test for software validity and there is no motivation for providing such a test in the Maitra invention.

The examiner's response to the above arguments was as follows: "Jablon shows the use of checksums in BIOS extensions that are programs that are loaded into memory for execution. The argument the [*sic*] Jablon makes for not using the checksums are in relation to viruses, which is not the subject of applicant's application (e.g. col. 5, lines 4-18)." 03/26/04 Office Action, p. 16, ¶ 92.

The examiner seems to be under the impression that all he has to do to establish the obviousness of claim 3 is to find a reference in which "checksum" is mentioned. He ignores all of the other words in claim 3 and those of claims 1 and 2 which provide the context of claim 3. The examiner's response is NOT a rebuttal to applicants' arguments for the nonobviousness of claims 3 and 28.

The examiner has not established *prima facie* obviousness of claims 3 and 28.

CLAIMS 4 AND 29

Claim 4 reads as follows:

4. *The method of claim 2 wherein a software package is assigned its own dedicated memory region, one of the tests for validity being whether the address returned for the software package's initialization procedure lies within its dedicated memory region.*

Apparatus claim 29 contains the same limitation as claim 4.

Neither of the cited references discloses the limitation of claim 4. The passage in Jablon et al. cited by the examiner (col. 6, lines 27-37) (03/26/04 Office Action, p. 5, ¶ 25) points out that methods for partitioning memory "generally allow trusted software to both enable and disable the protection mechanism for a given region of memory" and that the memory protection feature in the Jablon et al. invention only allows software control from unprotected to protected mode. Nothing is said about "one of the tests for validity being whether the address returned for the software package's initialization procedure lies within its dedicated memory region."

Since neither Maitra nor Jablon et al. disclose the limitations of claims 4 and 29, there is obviously no motivation for a person skilled in the art for incorporating such limitations in the Maitra invention.

Thus, Maitra and Jablon et al. in combination do not meet two of the basic criteria for establishing *prima facie* obviousness of claims 4 and 29. Neither of the references discloses applicants' address-returned test for software validity and there is no motivation for providing such a test in the Maitra invention.

The examiner's response to the above arguments was as follows:

03/26/04 Office Action, p. 17, ¶ 94 - "It would have been obvious . . . to combine Maitra and Jablon in order to test the validity of the software application using a one's complement [sic] checksum method."

Comment - The term "one's complement checksum" is not mentioned in claims 4 and 29. Neither reference discloses "a software package is assigned its own dedicated memory region" or "one of the tests of validity being whether the address returned for the software package's initialization procedure lies within its dedicated memory region." How could it be obvious to combine the references when neither reference discloses two of the limitations of the claims?

03/26/04 Office Action, p. 17, ¶ 96 - "Jablon discusses the method to allow the memory address space to be portioned and allow control over which software has access to individual regions of the memory."

Comment - This is not a disclosure of either of the claim 4/claim 29 limitations "a software package is assigned its own dedicated memory region" and "one of the tests of validity being whether the address returned for the software package's initialization procedure lies within its dedicated memory region."

03/26/04 Office Action, p. 18, ¶ 98 - "It would have been obvious . . . to combine Maitra and Jablon in order to have set memory partitions for each software package and the validity test being conducted within predefined memory regions."

Comment - If by "set memory partitions for each software package" the examiner means "a software package is assigned its own dedicated memory region", neither reference discloses this limitation. Neither of the claims include the limitation "the validity test being conducted within predefined memory regions." There is no motivation to combine two references when neither reference discloses the limitations of the claims.

The examiner has not established *prima facie* obviousness of claims 4 and 29.

CLAIMS 5 AND 30

Claim 5 reads as follows:

5. *The method of claim 4 wherein one of the tests is whether the address is returned within a predetermined time.*

Apparatus claim 30 contains the same limitation as claim 5.

Neither of the cited references discloses the limitations of claims 4 and 29 (see discussion above). The references also do not disclose the additional limitation of claims 5 and 30 which further limits the limitations of claims 4 and 29. The passage in Jablon et al. cited by the examiner (col. 7, lines 21-34) (03/26/04 Office Action, p. 5, ¶ 27) points out that "[o]ur invention assesses the integrity of the trusted software, to discover any corruption as the system starts. The combined hardware and software approach used here makes this method immune to software-only attack, and the security of this method does not depend on keeping secret the design details of either the software or the hardware." Nothing is said about "one of the tests [for validity] is whether the address is returned within a predetermined time."

Thus, Maitra and Jablon et al. in combination do not meet two of the basic criteria for establishing *prima facie* obviousness of claims 5 and 30. Neither of the references discloses applicants' address-returned within a predetermined time test for software validity and there is no motivation for providing such a test in the Maitra invention.

The examiner did not provide rebuttal to the arguments above in his 03/26/04 Office Action.

The examiner has not established *prima facie* obviousness of claims 5 and 30.

CLAIMS 18 AND 43

Claim 18 reads as follows:

18. *The method of claim 1 wherein safety-critical software is placed in one or more separate partitions thereby isolating the safety-critical software from non-safety-critical software.*

Apparatus claim 43 contains the same limitation as claim 18.

Neither of the cited references discloses the limitation of claim 18. The passage in Jablon et al. cited by the examiner (col. 6, lines 27-38) (03/26/04 Office Action, p. 6, ¶ 29) points out that methods for partitioning memory "generally allow trusted software to both enable and disable the protection mechanism for a given region of memory" and that the memory protection feature in the Jablon et al. invention only allows software control from unprotected to protected mode. Nothing is said about "safety-critical software . . . [being] placed in one or more separate partitions thereby isolating the safety-critical software from non-safety-critical software."

Since neither Maitra nor Jablon et al. disclose the limitations of claims 18 and 43, there is obviously no motivation for a person skilled in the art for incorporating such limitations in the Maitra invention.

Thus, Maitra and Jablon et al. in combination do not meet two of the basic criteria for establishing *prima facie* obviousness of claims 18 and 43. Neither of the references discloses applicants' safety-critical software isolation limitation and there is no motivation for providing such a feature in the Maitra invention.

The examiner's response to the above arguments was simply to reiterate that col. 6, lines 27-38, of Jablon et al. discloses the limitations of claim 18 and to state that "Jablon discusses a

protection mechanism for code to be placed [sic] in a protectable non-volatile storage area (e.g. Abstract). (03/26/04 Office Action, p. 18, ¶ 100). A more complete version of what is disclosed in the Jablon et al. Abstract is as follows:

"Programs and data comprising the system's trusted software, including all startup processes, are verified before being utilized. Methods to verify the trusted software use a hierarchy of both modification detection codes and public-key digital signature codes. The top-level codes are placed in a protectable non-volatile storage area, and are used by the startup program to verify the integrity of subsequent programs. A trusted initialization program sets a hardware latch to protect the codes in the non-volatile memory from being overwritten by subsequent untrusted programs."

Note that the "top-level codes", NOT an application, are placed "in a protectable non-volatile storage area." This is NOT a disclosure of the claim-18 method "wherein safety-critical software is placed in one or more separate partitions thereby isolating the safety-critical software from non-safety-critical software." There is nothing in the references concerning the storage of safety-critical software.

The examiner also argued that a person skilled in the art would be motivated "to combine Maitra and Jablon et al. in order to isolate different software packages in different partitions." (03/26/04 Office Action, p. 19, ¶ 102). A person skilled in the art may find motivation to incorporate certain features of the Jablon et al. invention in the Maitra invention, but these features will not include one relating to safety-critical software which is not mentioned by either reference.

The examiner has not established *prima facie* obviousness of claims 18 and 43.

CLAIMS 19 AND 44

Claim 19 reads as follows:

19. *The method of claim 1 wherein each of the plurality of software packages is assigned its own memory block, a software package being enableable to read data only from zero or more memory blocks associated with other software packages, the zero or more memory blocks readable by a software package being either predetermined or determined during execution of the software packages in accordance with a set of one or more rules.*

Apparatus claim 44 contains the same limitations as claim 19.

Neither of the cited references discloses the limitation of claim 19. The passage in Jablon et al. cited by the examiner (col. 8, lines 16-33) (03/26/04 Office Action, p. 6, ¶ 31) discloses a system wherein "during system initialization trusted software closes the latch to protect the memory, and thus prevent all subsequently run programs from reading and/or writing the security-relevant data during normal operation." This is not a disclosure of "a software package being enableable to read data only from zero or more memory blocks associated with other software packages, the zero or more memory blocks readable by a software package being either predetermined or determined during execution of the software packages in accordance with a set of one or more rules."

Since neither Maitra nor Jablon et al. disclose the limitations of claims 19 and 44, there is obviously no motivation for a person skilled in the art for incorporating such limitations in the Maitra invention.

Thus, Maitra and Jablon et al. in combination do not meet two of the basic criteria for establishing *prima facie* obviousness of claims 19 and 44. Neither of the references discloses

applicants' limitations and there is no motivation for providing such limitations in the Maitra invention.

The examiner's response to the above arguments was to argue that "the latch system discussed by Jablon and its associated rules allow software programs to access memory blocks for reading and writing during the startup procedure (e.g. col. 8, lines 20-34)." 03/26/04 Office Action, p. 20, ¶ 104.

A more complete version of the Jablon et al. passage cited by the examiner is as follows:

"One or more regions of non-volatile memory are provided, in which security-relevant data are stored. Access to a protectable memory region is controlled with a latch mechanism, such that the memory is always both readable and writable when the computer is first started, But during system initialization trusted software closes the latch to protect the memory, and thus prevent all subsequently run programs from reading and/or writing the security-relevant data during normal operation. Once closed, the latch can not be opened by software control. The latch is only re-opened when the system is restarted," Jablon et al., col. 8, lines 18-29.

Note the key disclosure shown in boldface. "Security-relevant DATA" (not an APPLICATION) is stored in "one or more regions of non-volatile memory". And this "security-relevant data" is what is readable or writable by an application. The claim language specifies that a software package be "enableable to read data only from zero or more memory blocks associated WITH OTHER SOFTWARE PACKAGES."

CLAIMS 20 AND 45

Claim 20 reads as follows:

20. *The method of claim 1 wherein each of the plurality of software packages is assigned its own memory block, a software package being enableable to write data only to zero or more memory blocks associated with other software packages, the zero or more memory blocks writeable by a software package being either predetermined or determined during execution of the software packages in accordance with a set of one or more rules.*

Apparatus claim 45 contains the same limitations as claim 20.

(NOTE: Claims 19 and 20 are essentially the same except that claim 19 has to do with readable access by one software package to the data associated with another software package while claim 20 has to do with writable access in the same situation. The argument given below for the nonobviousness of claim 20 is essentially a duplicate of the one given above in connection with claim 19.)

Neither of the cited references discloses the limitation of claim 20. The passage in Jablon et al. cited by the examiner (col. 8, lines 16-33) (03/26/04 Office Action, p. 7, ¶ 32) discloses a system wherein "during system initialization trusted software closes the latch to protect the memory, and thus prevent all subsequently run programs from reading and/or writing the security-relevant data during normal operation." This is not a disclosure of "a software package being enableable to write data only from zero or more memory blocks associated with other software packages, the zero or more memory blocks writeable by a software package being either predetermined or determined during execution of the software packages in accordance with a set of one or more rules."

Since neither Maitra nor Jablon et al. disclose the limitations of claims 19 and 44, there is obviously no motivation for a person skilled in the art for incorporating such limitations in the Maitra invention.

Thus, Maitra and Jablon et al. in combination do not meet two of the basic criteria for establishing *prima facie* obviousness of claims 19 and 44. Neither of the references discloses applicants' limitations and there is no motivation for providing such limitations in the Maitra invention.

The examiner's response to the above arguments was to argue that "the latch system discussed by Jablon and its associated rules allow software programs to access memory blocks for reading and writing during the startup procedure (e.g. col. 8, lines 20-34)." 03/26/04 Office Action, p. 20, ¶ 104. A more complete version of the Jablon et al. passage cited by the examiner is as follows:

"One or more regions of non-volatile memory are provided, in which security-relevant data are stored. Access to a protectable memory region is controlled with a latch mechanism, such that the memory is always both readable and writable when the computer is first started, But during system initialization trusted software closes the latch to protect the memory, and thus prevent all subsequently run programs from reading and/or writing the security-relevant data during normal operation. Once closed, the latch can not be opened by software control. The latch is only re-opened when the system is restarted," Jablon et al., col. 8, lines 18-29.

Note the key disclosure shown in boldface. "Security-relevant DATA" (not an APPLICATION) is stored in "one or more regions of non-volatile memory". And this "security-relevant data" is what is readable or writable by an application. The claim language specifies that a software package be "enableable to read data only from zero or more memory blocks associated WITH OTHER SOFTWARE PACKAGES."

**III. WHETHER CLAIMS 10, 12, 35, AND 37 ARE UNPATENTABLE
UNDER 35 U.S.C. § 103(A) IN VIEW OF MAITRA (U.S. 5,623,647) AND
REZNAK (U.S. 6,223,201).**

CLAIMS 10 AND 35

Claim 10 reads as follows:

10. *The method of claim 1 wherein a software package includes background tasks as well as foreground tasks, the background tasks being performed after the foreground tasks have been completed.*

Apparatus claim 35 contains the same limitations as claim 10.

Neither of the cited references discloses the limitation of claim 10. The passage in Reznak cited by the examiner (col. 1, lines 25-32) (03/26/04 Office Action, p. 7, ¶ 36) discloses (1) cooperative multitasking operating systems wherein background tasks are granted processing time only during idle periods of the foreground tasks and (2) time-slice multitasking operating systems wherein processing time is allocated to each task, be it foreground or background, in round-robin fashion or based upon task priority. Neither the cooperative multitasking operating system nor the time-slice multitasking operating system discloses applicants' limitation "the background tasks being performed after the foreground tasks have been completed." The cooperative multitasking operating system is disclosed as performing background tasks during the idle periods between foreground tasks. The time-slice multitasking operating system is disclosed as performing tasks which may be either foreground or background in a round-robin fashion or based upon task priority.

Since neither Maitra nor Reznak disclose the limitations of claims 10 and 35, there is obviously no motivation for a person skilled in the art for incorporating such limitations in the Maitra invention.

Thus, Maitra and Reznak in combination do not meet two of the basic criteria for establishing *prima facie* obviousness of claims 10 and 35. Neither of the references discloses applicants' limitations and there is no motivation for providing such limitations in the Maitra invention.

The examiner's response to the above arguments was to cite two passages from Reznak (03/26/04 Office Action, p. 22, ¶ 112), the first being as follows :

"Cooperative multitasking operating system classify tasks as either foreground or background tasks. Background tasks are granted processing time only during idle periods of the foreground tasks, such as periods when the foreground tasks are performing input and output operations. Time-slice multitasking operating systems, in contrast, allocate processing time (i.e., time slices) to each task in round robin fashion or based upon task priority." Col. 1, lines 25-32.

This is not a disclosure of "the background tasks being performed after the foreground tasks have been completed."

The second passage from Reznak cited by the examiner is as follows:

"In response to these deficiencies in conventional multitasking operating systems, multitasking operating systems have recently been developed which include scheduling facilities that utilize processing time estimates provided by tasks prior to dispatch to allocate processing resources to the tasks." Col. 1, lines 51-56

This passage has nothing to do with the subject matter of claim 10.

CLAIMS 12 AND 37

Claim 12 reads as follows:

12. *The method of claim 10 wherein the software package causes the power utilized in executing the software package to be minimized after completion of the background tasks.*

Apparatus claim 37 contains the same limitations as claim 12.

The only reference cited by the examiner for the obviousness of claims 12 and 37 is Maitra. The passage cited by the examiner (col. 2, lines 39-48) (03/26/04 Office Action, p. 8, ¶ 37) summarizes the Maitra invention as a method and apparatus for operating a microprocessor which reduces the power consumption and heat dissipation of the microprocessor. Maitra accomplishes this result by tailoring the clock speed of the microprocessor to the computing requirements of the application being executed (col. 2, lines 43-59). Maitra says nothing about the power utilized in executing the software package being minimized after completion of the background tasks. Maitra never discusses foreground and background tasks.

Since Maitra does not disclose the limitations of claims 12 and 37, there is obviously no motivation for a person skilled in the art for incorporating such limitations in the Maitra invention.

Thus, Maitra does not meet two of the basic criteria for establishing *prima facie* obviousness of claims 12 and 37. Maitra does not disclose applicants' limitations and there is no motivation for providing such limitations in the Maitra invention.

The examiner's response to the above arguments was to cite another passage from Maitra (03/26/04 Office Action, p. 23, ¶ 116):

"The present invention relates to a method and apparatus for operating a microprocessor which reduces the power consumption and heat dissipation of the microprocessor without

any user perceived degradation in performance. This method of operating the microprocessor typically includes determining the type of application that will be run by the microprocessor during a specific time quantum. This information is usually available in a computer's operating system's task scheduler. After the application is determined, the computing requirement of that process is determined." Col. 2, lines 39-48.

This is not a disclosure of "the power utilized in executing the software package to be minimized after completion of the background tasks."

IV. WHETHER CLAIMS 11 AND 36 ARE UNPATENTABLE UNDER 35 U.S.C. § 103(A) IN VIEW OF MAITRA (U.S. 5,623,647), REZNAK (U.S. 6,223,201), AND DAVIDSON ET AL. (U.S. 6,292,934).

CLAIMS 11 AND 36

Claim 11 reads as follows:

11. *The method of claim 10 wherein a background task is an infinite loop.*

Apparatus claim 36 contains the same limitation as claim 11.

The examiner cites Davidson et al., col. 17, lines 32-58, (03/26/04 Office Action, p. 8, ¶ 40) as disclosing a background task which is an infinite loop.

The infinite loop of Davidson et al. is not a background task. It is a result of replacing a conditional branch instruction from a live code block to a dead code or rarely-executed code block by a conditional branch-to-self instruction, thereby saving memory storage space. The occurrence of the infinite loop that results from the execution of the branch-to-self instruction is detected by a monitor process which causes the program to branch to the rarely executed code. Davidson et al.,

col. 17, lines 3-14. In other words, the occurrence of an infinite loop is an indication that the program should be redirected to executing the rarely executed code. The occurrence of an infinite loop as a means of redirecting the execution of a program is not a disclosure of applicants' background task infinite loop.

Thus, Maitra, Reznak, and Davidson et al. in combination do not meet two of the basic criteria for establishing *prima facie* obviousness of claims 11 and 36. The combination of references does not disclose applicants' claim limitations and there is no motivation for providing such limitations in the Maitra invention.

The examiner responded to the above argument by repeating the citation from Davidson et al. given above.

**V. WHETHER CLAIMS 13, 14, 38, AND 39 ARE UNPATENTABLE
UNDER 35 U.S.C. § 103(A) IN VIEW OF MAITRA (U.S. 5,623,647) AND
YEN (U.S. 6,381,694).**

CLAIMS 13 AND 38

Claim 13 reads as follows:

13. *The method of claim 1 wherein a failure in the execution of a software package causes information to be logged in a failure log.*

Apparatus claim 38 contains the same limitation as claim 13.

The examiner cites Yen, col. 6, lines 49-59, (03/26/04 Office Action, p. 8, ¶ 44) as disclosing a failure in the execution of a software package causing information to be logged in a failure log.

The cited passage discloses a recovery system which only logs information relating to a startup failure. Yen defines a startup failure as follows: "If however, the operating system file cannot be located in the main volume, or if certain files are found to be corrupted, a startup failure occurs." Yen, col. 3, lines 48-50. Failure in startup of the operating system of a computer is not a failure to execute a software package that is only attempted after the operating system is operating.

Thus, Maitra and Yen in combination do not meet two of the basic criteria for establishing *prima facie* obviousness of claims 13 and 38. The combination of references does not disclose applicants' claim limitation and there is no motivation for providing such a limitation in the Maitra invention.

The examiner responded to the above argument by repeating the citation from Yen given above. 03/26/04 Office Action, p. 24, ¶ 122.

CLAIMS 14 AND 39

Claim 14 reads as follows:

14. *The method of claim 13 wherein a failure in execution is linked to the software package that caused the failure.*

Apparatus claim 39 contains the same limitation as claim 14.

The examiner cites Yen, col. 5, lines 1-16, and Figs. 3 and 9, (03/26/04 Office Action, p. 9, ¶ 46) as disclosing a failure in the execution of a software package being linked to the software package that caused the failure.

The Yen patent is for a system for recovering from certain types of system software startup problems whereby if an error is detected that would normally result in a startup failure, the

computer's startup routine branches to an alternate startup application. Yen, Abstract. Yen defines a startup failure as follows: "If however, the operating system file cannot be located in the main volume, or if certain files are found to be corrupted, a startup failure occurs." Yen, col. 3, lines 48-50. The passage cited by the examiner only discloses the options a user has available when a startup failure occurs. There is no disclosure pertaining to multi-tasking computer systems and failures in execution of one software package caused by another software package which would occur after the computer operating system has been started.

Maitra and Yen in combination do not meet two of the basic criteria for establishing *prima facie* obviousness of claims 14 and 39. The combination of references does not disclose applicants' claim limitation and there is no motivation for providing such a limitation in the Maitra invention.

The examiner responded to the above argument by repeating the citation from Yen given above. 03/26/04 Office Action, p. 25, ¶ 126.

**VI. WHETHER CLAIMS 15-17 AND 40-42 ARE UNPATENTABLE UNDER
35 U.S.C. § 103(A) IN VIEW OF MAITRA (U.S. 5,623,647), YEN (U.S.
6,381,694), AND HARPER ET AL. (U.S. 6,629,266).**

CLAIMS 15 AND 40

Claim 15 reads as follows:

15. *The method of claim 13 wherein quality of performance in executing a software package is represented by one or more performance-quality parameters, values of the one or more performance-quality parameters being determined from the information logged in a failure log, the execution of a software package being subject to a plurality of execution options, an execution option being selected on the basis of one or more performance-quality parameter values.*

Apparatus claim 40 contains the same limitation as claim 15.

The examiner cites Harper et al. as disclosing the limitations of claim 15. 03/26/04 Office Action, p. 10, ¶ 49. The Harper et al. invention is a method and system for doing the following (col. 3, lines 1-59):

- ♦ learning how to predict outage of software system;
- ♦ predicting software outages;
- ♦ rejuvenating software; and
- ♦ predicting impending resource exhaustion and aging.

None of these tasks has anything to do with the limitations of claim 15. Nothing is disclosed concerning "quality of performance". Nothing is disclosed concerning "performance-quality parameters". Nothing is disclosed concerning the determination of the values of the "performance-quality parameters" from information logged in a failure log. Nothing is disclosed concerning the execution of a software package being subject to a plurality of "execution options". And nothing is disclosed concerning an "execution option" being selected on the basis of one or more "performance-quality parameter" values.

The examiner cites a 15-line passage from Harper et al. (col. 13, lines 35-50) as disclosing the entirety of the limitations of claim 15. Instead, the passage discusses the gathering of information for purposes of software rejuvenation and resource exhaustion—items which have nothing to do with applicants' claim 15.

Maitra and Harper et al. in combination do not meet two of the basic criteria for establishing *prima facie* obviousness of claims 15 and 40. The combination of references does not disclose

applicants' claim limitations and there is no motivation for providing such limitations in the Maitra invention.

The examiner responded to the above arguments by stating that "Harper teaches that DAC unit receives performance parameters either from the operating system or from a log file and send these parameters to the XTALK module. The chain of resources is a simple circular linked list that contains description of the parameters that are to be monitored for resource exhaustion. Figures 16 and 17 show some of the performance parameters (e.g. col 13, lines 35-67 and col. 14, lines 1-35)." 03/26/04 Office Action, p. 26, ¶ 130.

The DAC unit that the examiner refers to is Operating System specific since it "polls the data directly from the Operating System." Harper et al., col. 13, lines 40-42. The data obtained by the DAC unit is therefore not useful in determining quality of performance of the software packages in a multi-tasking computer system. The examiner seems to find the "chain of resources . . . that are to be monitored for resource exhaustion" to be of interest in determining quality of performance of the software packages in a multi-tasking computer system. However, the "chain of resources . . . that are to be monitored for resource exhaustion" traces back to the computer operating system (Harper et al., col. 13, line 35 - col. 14, line 9) and is not helpful except in a second-hand way in determining quality of performance of the software packages in a multi-tasking computer system.

In short, there is no merit in the examiner's rebuttal of the arguments for the nonobviousness of claim 15.

CLAIMS 16 AND 41

Claim 16 reads as follows:

16. *The method of claim 15 wherein the plurality of execution options are user configurable.*

Apparatus claim 41 contains the same limitation as claim 16.

Since Harper et al. does not disclose "execution options" (see discussion above under the CLAIMS 15 AND 40 heading), a disclosure by Harper et al. that the "execution options" are user configurable is inconceivable. The examiner, however, cites Harper et al.'s Fig. 5 as being such a disclosure. 03/26/04 Office Action, p. 10, ¶ 51. Fig. 5 has to do with symptom-based software rejuvenation and has nothing to do with "execution options" that are user configurable.

Maitra and Harper et al. in combination do not meet two of the basic criteria for establishing *prima facie* obviousness of claims 16 and 41. The combination of references does not disclose applicants' claim limitation and there is no motivation for providing such a limitation in the Maitra invention.

The examiner did not respond to the above arguments. 03/26/04 Office Action, p. 26, ¶ 132 (pertains to claim 15), p.27, ¶ 133 (pertains to claim 23).

CLAIMS 17 AND 42

Claim 17 reads as follows:

17. *The method of claim 15 wherein performance-quality parameters include the number of failures and/or the rate of failures for one or more classes of failures recorded in a software package's failure log.*

Apparatus claim 42 contains the same limitation as claim 17.

Since Harper et al. does not disclose "performance-quality parameters" pertaining to software packages in a multi-tasking computer system (see discussion above under the CLAIMS 15 AND 40), an enumeration by Harper et al. of some of the "performance-quality parameters" seems unlikely. The examiner, however, cites col. 2, lines 32-35, of Harper et al. as being such a disclosure. 03/26/04 Office Action, p. 10, ¶ 53. However this passage simply states that in a co-pending application "it was described how to periodically rejuvenate all or part of a software system to reduce its failure rate to its initial, lower lever, based on time." This is not a disclosure of the claim-17 limitations. The examiner also cited Harper et al.'s Fig. 1 which shows a plot of software failure rate vs. time. This also is not a disclosure of the claim-17 limitations.

Maitra and Harper et al. in combination do not meet two of the basic criteria for establishing *prima facie* obviousness of claims 17 and 42. The combination of references does not disclose applicants' claim limitation and there is no motivation for providing such a limitation in the Maitra invention.

The examiner did not respond to the above arguments. 03/26/04 Office Action, p. 26, ¶ 132 (pertains to claim 15), p.27, ¶ 133 (pertains to claim 23).

VII. WHETHER CLAIMS 23 AND 48 ARE UNPATENTABLE UNDER 35 U.S.C. § 103(A) IN VIEW OF MAITRA (U.S. 5,623,647) AND POTTER ET AL. (U.S. 5,014,327).

CLAIMS 23 AND 48

Claim 23 reads as follows:

23. *The method of claim 1 wherein one or more of the plurality of software packages are independently compiled, linked, and loaded.*

Apparatus claim 48 contains the same limitation as claim 23.

The examiner cites Potter et al. (col. 7, lines 40-57) (03/26/04 Office Action, p. 11, ¶ 56) as disclosing the limitation of claim 23. Potter et al. discloses a pattern recognition system 10 which recognizes patterns by means of the pattern recognition software package consisting of application 11 and data collectors and data manipulation portions 12. Potter et al., col. 7, lines 23-39. The passage cited by the examiner states that the pattern recognition software package is "written in source code which is compiled, linked and loaded by the Operating System 13, and which may incorporate subroutines included in the Run Time Library 15." This is not a disclosure of a method for repetitively executing a plurality of software packages (Claim 1) "wherein one or more of the plurality of software packages are independently compiled, linked, and loaded." The disclosure that an application consisting of two linked tasks is compiled, linked, and loaded by the operating system of the computer on which it will run is not a disclosure of a multi-tasking computer system wherein on or more software packages are independently compiled, linked, and loaded.

Maitra and Potter et al. in combination do not meet two of the basic criteria for establishing *prima facie* obviousness of claims 23 and 48. The combination of references does not disclose applicants' claim limitation and there is no motivation for providing such a limitation in the Maitra invention.

The examiner responded to the above argument by repeating the citation from Potter et al. given above. 03/26/04 Office Action, p. 27, ¶ 134.

VIII. WHETHER CLAIMS 24 AND 49 ARE UNPATENTABLE UNDER 35 U.S.C. § 103(A) IN VIEW OF MAITRA (U.S. 5,623,647) AND CIRCELLO ET AL. (U.S. 5,761,491).

CLAIMS 24 AND 49

Claim 24 reads as follows:

24. *The method of claim 1 wherein a software package has its own stack, the software package's stack being selected prior to executing the software package.*

Apparatus claim 49 contains the same limitation as claim 24.

The examiner cites Circello et al. as disclosing the limitations of claim 24 (03/26/04 Office Action, p. 11, ¶ 59). Circello et al. discloses "a data processor and method of operating a data processing system in which a single system stack pointer may be used to create records of both system and user stack operations when hardware support for alignment of such stack operands is optional." (Col. 3, lines 6-10) This is not a disclosure of a method for repetitively executing a plurality of software packages (Claim 1) "wherein a software package has its own stack, the software package's stack being selected prior to executing the software package."

The examiner points specifically to Circello et al., col. 1, lines 35-51, which mentions that "user programs routinely manipulate the contents of the stack pointer which defines the top of the stack." However, the mere mention of "user programs" and "stack" in the same sentence does not disclose a multi-tasking computer structure wherein one or more of the software packages being executed may have their own stacks. Nor is it a disclosure of the software package's stack being selected prior to executing the software package.

Maitra and Circello et al. in combination do not meet two of the basic criteria for establishing *prima facie* obviousness of claims 24 and 49. The combination of references does not disclose applicants' claim limitation and there is no motivation for providing such a limitation in the Maitra invention.

The examiner responded to the above argument by repeating the citation from Circello et al. given above. 03/26/04 Office Action, p. 28, ¶ 138.

CONCLUSIONS

Appellants' invention is a method and apparatus for repetitively executing a plurality of software packages at a plurality of rates utilizing a common set of computational resources and predictive scheduling. Predictive scheduling is accomplished by generating a sequence of time intervals for each of the plurality of software packages and executing each of the plurality of software packages during each time interval in its sequence of time intervals.

The examiner rejected all of appellants' claims. The cornerstone of the examiner's rejections is the Maitra patent which discloses a just-in-time scheduling technique whereby a list of software packages is maintained, and the software packages on the list are executed in sequence. When the execution of one software package ends, the execution of the next software package on the list begins. There is no way of predicting when any of the software packages on the list will begin execution in the future.

A key step in appellants' predictive scheduling method is "generating a sequence of time intervals for each of the plurality of software packages." Maitra discloses (col. 6, lines 20-34), in the words of the examiner, "a multi-tasking environment whereby several applications are run simultaneously [and] [t]he operation of each software would not affect the operation of other software programs running at same or different speeds." The examiner argues that this is a disclosure of "generating a sequence of time intervals for each of the plurality of software packages."

Clearly, the examiner's contention is incorrect. There is nothing in the Maitra passage cited by the examiner which even implies the step of "generating a sequence of time intervals for each of the plurality of software packages." In fact, there is nothing in Maitra or any of the other references

Application Number: 09/821,537

P573C

Art Unit: 2154

cited by the examiner which disclose the limitations of any of the 49 claims which have been rejected
by the examiner.

The Board should reverse the examiner's rejection of the 49 claims subject to appeal.

APPENDIX

1. A method for repetitively executing a plurality of software packages at one or more rates, utilizing a common set of computational resources, the method comprising the steps:

generating a sequence of time intervals for each of the plurality of software packages, the time intervals belonging to one software package not overlapping the time intervals belonging to any other of the plurality of software packages;

executing a plurality of software packages, each software package being executed during the time intervals of its sequence of time intervals.

2. The method of claim 1 wherein the plurality of software packages of the "executing" step includes only valid software packages, the method further comprising the step:

utilizing one or more tests to identify the software packages that are valid.

3. The method of claim 2 wherein one of the tests for validity is a one's complement checksum test of a software package's program memory.

4. The method of claim 2 wherein a software package is assigned its own dedicated memory region, one of the tests for validity being whether the address returned for the software package's initialization procedure lies within its dedicated memory region.

5. The method of claim 4 wherein one of the tests is whether the address is returned within a predetermined time.

6. The method of claim 2 wherein a software package is assigned its own dedicated memory region, the software package's dedicated memory region including a stack memory region and/or a heap memory region, one of the tests for validity being whether the stack memory range and/or the heap memory range assigned during the execution of the software package's initialization

procedure and the various associated entry points lies within the software package's dedicated memory region.

7. The method of claim 6 wherein one of the tests is whether the stack memory range and/or the heap memory range and the various associated entry points are returned within a predetermined time.

8. The method of claim 1 wherein a software package is assigned its own dedicated memory region.

9. The method of claim 8 wherein the software package's dedicated memory region includes a stack memory region, a software package's stack residing in the software package's stack memory region.

10. The method of claim 1 wherein a software package includes background tasks as well as foreground tasks, the background tasks being performed after the foreground tasks have been completed.

11. The method of claim 10 wherein a background task is an infinite loop.

12. The method of claim 10 wherein the software package causes the power utilized in executing the software package to be minimized after completion of the background tasks.

13. The method of claim 1 wherein a failure in the execution of a software package causes information to be logged in a failure log.

14. The method of claim 13 wherein a failure in execution is linked to the software package that caused the failure.

15. The method of claim 13 wherein quality of performance in executing a software package is represented by one or more performance-quality parameters, values of the one or more performance-quality parameters being determined from the information logged in a failure log, the

execution of a software package being subject to a plurality of execution options, an execution option being selected on the basis of one or more performance-quality parameter values.

16. The method of claim 15 wherein the plurality of execution options are user configurable.

17. The method of claim 15 wherein performance-quality parameters include the number of failures and/or the rate of failures for one or more classes of failures recorded in a software package's failure log.

18. The method of claim 1 wherein safety-critical software is placed in one or more separate partitions thereby isolating the safety-critical software from non-safety-critical software.

19. The method of claim 1 wherein each of the plurality of software packages is assigned its own memory block, a software package being enableable to read data only from zero or more memory blocks associated with other software packages, the zero or more memory blocks readable by a software package being either predetermined or determined during execution of the software packages in accordance with a set of one or more rules.

20. The method of claim 1 wherein each of the plurality of software packages is assigned its own memory block, a software package being enableable to write data only to zero or more memory blocks associated with other software packages, the zero or more memory blocks writeable by a software package being either predetermined or determined during execution of the software packages in accordance with a set of one or more rules.

21. The method of claim 1 wherein an executive software package enforces the discipline that each software package executes only during the time intervals of its sequence of time intervals, the executive software package determining when the execution of a software package

extends into a time interval belonging to the sequence of time intervals assigned to another software package and performs a remedial action.

22. The method of claim 1 wherein the presence of those software packages that are present is detected.

23. The method of claim 1 wherein one or more of the plurality of software packages are independently compiled, linked, and loaded.

24. The method of claim 1 wherein a software package has its own stack, the software package's stack being selected prior to executing the software package.

25. Apparatus for practicing the method of claim 1.

26. Apparatus for repetitively executing a plurality of software packages at a plurality of rates, the apparatus comprising:

a means for generating a sequence of time intervals for each of the plurality of software packages, the time intervals belonging to one software package not overlapping the time intervals belonging to any other of the plurality of software packages;

a means for executing a plurality of software packages, each software package being executed during the time intervals of its sequence of time intervals.

27. The apparatus of claim 26 wherein the plurality of software packages executed by the "executing" means includes only valid software packages, the apparatus further comprising:

a means for utilizing one or more tests to identify the software packages that are valid.

28. The apparatus of claim 27 wherein one of the tests for validity is a one's complement checksum test of a software package's program memory.

29. The apparatus of claim 27 wherein a software package is assigned its own dedicated memory region, one of the tests for validity being whether the address returned for the software package's initialization procedure lies within its dedicated memory region.

30. The apparatus of claim 29 wherein one of the tests is whether the address is returned within a predetermined time.

31. The apparatus of claim 27 wherein a software package is assigned its own dedicated memory region, the software package's dedicated memory region including a stack memory region and/or a heap memory region, one of the tests for validity being whether the stack memory range and/or the heap memory range assigned during the execution of the software package's initialization procedure and the various associated entry points lies within the software package's dedicated memory region.

32. The apparatus of claim 31 wherein one of the tests is whether the stack memory range and/or the heap memory range and the various associated entry points are returned within a predetermined time.

33. The apparatus of claim 26 wherein a software package is assigned its own dedicated memory region.

34. The apparatus of claim 33 wherein the software package's dedicated memory region includes a stack memory region, a software package's stack residing in the software package's stack memory region.

35. The apparatus of claim 26 wherein a software package includes background tasks as well as foreground tasks, the background tasks being performed after the foreground tasks have been completed.

36. The apparatus of claim 35 wherein a background task is an infinite loop.

37. The apparatus of claim 35 wherein the software package causes the power utilized in executing the software package to be minimized after completion of the background tasks.

38. The apparatus of claim 26 wherein a failure in the execution of a software package causes information to be logged in a failure log.

39. The apparatus of claim 38 wherein a failure in execution is linked to the software package that caused the failure.

40. The apparatus of claim 38 wherein quality of performance in executing a software package is represented by one or more performance-quality parameters, values of the one or more performance-quality parameters being determined from the information logged in a failure log, the execution of a software package being subject to a plurality of execution options, an execution option being selected on the basis of one or more performance-quality parameter values.

41. The apparatus of claim 40 wherein the plurality of execution options are user configurable.

42. The apparatus of claim 40 wherein performance-quality parameters include the number of failures and/or the rate of failures for one or more classes of failures recorded in a software package's failure log.

43. The apparatus of claim 26 wherein safety-critical software is placed in one or more separate partitions thereby isolating the safety-critical software from non-safety-critical software.

44. The apparatus of claim 26 wherein each of the plurality of software packages is assigned its own memory block, a software package being enableable to read data only from zero or more memory blocks associated with other software packages, the zero or more memory blocks readable by a software package being either predetermined or determined during execution of the software packages in accordance with a set of one or more rules.

45. The apparatus of claim 26 wherein each of the plurality of software packages is assigned its own memory block, a software package being enableable to write data only to zero or more memory blocks associated with other software packages, the zero or more memory blocks writeable by a software package being either predetermined or determined during execution of the software packages in accordance with a set of one or more rules.

46. The apparatus of claim 26 wherein an executive software package enforces the discipline that each software package executes only during the time intervals of its sequence of time intervals, the executive software package determining when the execution of a software package extends into a time interval belonging to the sequence of time intervals assigned to another software package and performs a remedial action. 30

47. The apparatus of claim 26 wherein the presence of those software packages that are present is detected.

48. The apparatus of claim 26 wherein one or more of the plurality of software packages are independently compiled, linked, and loaded.

49. The apparatus of claim 26 wherein a software package has its own stack, the software package's stack being selected prior to executing the software package.